

# 6 DOF Nonlinear AUV Simulation Toolbox

Xiaodong Chen, Dave Marco, Sam Smith, Edgar An, K. Ganesan, Tony Healey

Email: xdchen@oe.fau.edu

## Abstract:

This paper describes the organization of 6 DOF nonlinear autonomous underwater vehicle (AUV) simulation toolbox, which is currently under development for the Ocean Explorer (OEX) series AUVs developed at Florida Atlantic University. This software development is part of 5-year ONR MURI effort of which its goal is to develop innovative tools and methodologies for the control of complex nonlinear dynamic systems. The purpose of this software simulation is to supply a flexible 3D-simulation platform for motion visualization, in-lab debugging and testing of mission-specific strategies as well as those related to C3 purposes. This software is currently jointly developed by the Ocean Engineering Department at Florida Atlantic University and Naval Postgraduate School for the FAU OEX and NPS Phoenix AUVs.

## Introduction

The new generation multipurpose autonomous underwater vehicles (AUV) such as Ocean Explorer are modular designed [Smith] in order to cut time and cost for vehicle reconfiguration. A flexible 3D-simulation platform is desired to visualize testing of AUV components before porting to vehicle meanwhile platform can also be used for mission replay. Comparing to the most available simulation software, advantage for using this software architecture, which is almost identical to the OEX software running on real-time operation system VxWorks, is based on behavioral decomposition methodology, where each behavior is implemented as an independent Unix process. A behavior is viewed as an independent conceptual task that cooperates with other behaviors in the system to perform a given mission. The nearly identical structure gives the user great flexibility that program tested on the simulation platform can be directly ported with few or no modification onto the OEX and this is expected to reduce the development cycle by minimizing the difference between the actual system and simulation environment. Shared memory is used as the backbone of data communication among processes instead of conventional message passing. Access control of

shared memory and data synchronization is achieved by using semaphores. Simulation programs are written in C and C++. In particular, Motif is used to design GUI and Open Inventor is used to create 3D models. Current implemented modules include a nonlinear dynamic model for the OEX, shared memory and semaphore manager tools, shared memory monitor, data logger, fuzzy sliding mode pitch controller and 3D graphics interface for displaying real-time or post-processed data. Additional modules which include vehicle hydrodynamics model, thruster model, wave and current models, sensor model, an annotated map for bottom terrain will also be implemented within the project duration.

## Model of Ocean Explorer

The Ocean Explorer AUV (Figure 1) is 7.14 feet long and 21 inches in maximum diameter. Weight in air is 714.2lbs and displaced weight is 716.7lbs. Hull volume is 11.1931ft<sup>3</sup>. It is designed with

- a teardrop shaped fiberglass hull based on a modified version of the Gertler Series 58 Model 4154-body shape.
- aft-mounted cruciform control surfaces.
- a 3-bladed propeller 18 inches in diameter.

- intelligent Ni-Cd battery packs. The battery packs can supply up to 12 hours of continuous missions at 3-knot cruising speed.
- main computer and electronics board (MC68030 at 50 MHz on the VME bus). Each of the components is embedded with a LonWorks Neuron node, and the control communication is achieved via LonTalk protocol.
- sensors include Watson AHRS-C302RS (3-axis acceleration, angles and rates), SIMRAD mesotech 809 (altitude), Druck PTX 1649 (water depth), Sonic Speed (water speed), DGPS, LBL and USBL positioning system.

The hydrodynamic and maneuvering models of OEX have been developed [Humphreys]. Vehicle equations and notation can be referred from [Fossen]. Figure 2 shows an open loop time history of the heading angle and yaw rate while rudder angle was set to 0.1 radius. (Assuming a constant propeller force). More work on developing a suitable thruster model is needed.

### Shared memory and Semaphore

The critical aspect of multiprocessing programming is the communication among processes. Shared memory is chosen as the way of communication because it is fast and effective [Ganesan]. Semaphore is used to control the access of shared memory to avoid the conflict and for synchronization problem. The normal procedure for any process to access the shared memory is to acquire the related semaphore first, then access the data and then give up the semaphore. Shared memory is implemented as a multiple units' data storage where related variables are carefully grouped. If the shared memory is considered as a single unit, most of processes have to wait for a long time before they get the semaphore. If each variable is assigned its own semaphore, synchronization can not be guaranteed or deadlock may occur. The number of semaphores within a group is also limited on Unix. The structure of shared memory must be very carefully chosen. A simple text file "shmem.in" with C-like language is used to make it easy to design or reconstruct the required shared memory structure such as the variable

name, type, initial value, unit, sampling rate and etc. A sample from "shmem.in" is listed below:

```
// Sync
AuvMotion          // written by StateMgr
{
    head "deg" double = 0.0; 8 IfChanged;
    roll "deg" double = 0.0; 8 IfChanged;
    pitch "deg" double = 0.0; 8 IfChanged;
    yawRate "deg/s" double = 0.0;
    rollRate "deg/s" double = 0.0;
    pitchRate "deg/s" double = 0.0;
    xAcc "G" double = 0.0;
    yAcc "G" double = 0.0;
    zAcc "G" double = 0.0;
};
```

Here a unit which includes head, roll, pitch etc. called 'AuvMotion' in shared memory is defined where "head" is a "double" type variable with initial value 0.0. The unit of head is degree. The sampling frequency is 8Hz. The logger strategy is head value will be logged only if it is changed. Two filter programs are supplied to convert "shmem.in" to C source code, one for VxWorks and one for Unix.

All processes can only interact with shared memory via dedicated function call to prevent the incorrect operation on shared memory. To minimize the time for a process of keeping semaphore, two operations are usually used by special functions, "SMGlobalToLocal" and "SMLocalToGlobal", which are used for process to read from and write to shared memory. (Figure 3).

A shared memory and semaphore manage tool is designed for users convenience. Users can use the tool to create or delete shared memory and semaphores, can get variable information such as variable values and relative semaphore number, can set special semaphore values to block or release the access to particular part of shared memory and can reinitialize shared memory. The Motif version manage tool is also available.

### Arbiter and Manager

The advantage of OEX behavior-based structure design as compared with hierarchical architecture is that interaction among processes is highly modular and thus processes can be added or removed without much interfering with others,

thereby minimizing software rewrite. However multi-processes have multi-output. An arbiter is designed to make a final decision for its given multiple input behaviors. There are several different types of arbiters, such as Boolean arbiter, (fuzzy) weighted average arbiter and fuzzy constrains. The difference between fuzzy constrain type arbiter and others is its input and output are fuzzy sets instead of simple values. The final result will be defuzzified.

Usually an arbiter take the confidence, importance and variable value as inputs and generate an output with its confidence and importance. Confidence represents a degree of accuracy of input data and importance initially is a user-defined value.

In order to make behaviors well organized, manager is used to manager related behaviors and arbiters and is used to schedule the start time, priority of processes and decide the final output by using arbiters. There are three different types of managers: synchronous, asynchronous and event manager. The event manager is used for handle condition-based execution. The advantage for using manager is that it can be dynamically reconfigured

For example in Figure 4, manager manages two controllers and one arbiter. Two controllers are depth and altitude controllers. An arbiter is used to combine the commanded pitch from both controllers and open loop pitch then generates final pitch angle.

## Mission Plan

Mission plan is a plain text file with English style language. Below is a typical mission file:

```
// set initial point
Set Origin 26 22.22 N 80 6.27 W
// start state manager with priority 70
Start Mgr StateMgr 70
Start Mgr HeadingMgr 80
//disable the surface safety feature
Need NoSurfaceSafety
Start Motor
clear criteria
set criteria plane -5
set speed 1.5
set depth 8.0
goto xy 0 -1800
set depth 0
goto xy 200 -2200
//end mission
clear criteria
```

```
set rpm()
set torque()
set direction()
Go
Stop motor
Stop mission
```

For a typical mission listed above, AUV was first commanded go south 1800m with speed 1.5m/sec and depth 8.0m, then come to surface for GPS fix and go south 400m. The required manager is state manager and heading manager which run at priority 70 and 80. "Set criteria" is used to set the criteria of the completion of a set point or way point command. "Go" means to start the mission.

Simulation software shares the same mission plan file as vehicle. So mission status can be easily relayed to simulation platform.

## Monitor and Data Logger

Monitor is an independent process with Motif GUI used to monitor and change the values of shared memory variables. Monitor can also be used to show the real vehicle motion dynamically via client server communication. One advantage of monitor is you can change the shared memory variables at any time. So it can be used for "hardware in loop" simulation.

An editable monitor parameter file is generated by filter program when you generated the source code for shared memory. Users can select their desired monitor variables.

Data logger is used to record the variable values along with the updating time for further use. Variables can be logged at a user given sampling frequency or be logged if it is changed. A parameter text file for logger is also supplied for user connivance. Logger data can be rewritten to shared memory for mission replay on graphical platform. Extracting data from logger file is achieved by a supplied program. Data analysis package written in MatLab is also available.

## Controller

TSK type fuzzy controllers were widely used to control vehicle motion such as heading, pitching and depth because of their robustness. For an example, the pitch controller takes in pitch and pitch rate as inputs and produces a desired stern plane angle which was then fed to the vehicle. The detailed description of the fuzzy controller can be found in [An]. Fuzzy sliding

mode controller is currently under being developed and tested (Figure 5) .

As parameters of controllers are stored in shared memory, the performance of controller can be dynamically changed. With the help of simulation platform users can easily tune the parameters of controller.

### 3D Graphics

3D graphics is used to help user visualize AUV motion. OpenGL, Open Inventor, Performer are three most widely used developing tools. OpenGL is industry standard and platform independent. It provides C, FORTRAN, Ada API. But it is a low-level graphics programming tools. It is hard to develop complex graphic application. Performer is easy to use and provides C, C++ API. But it is platform dependent. Open Inventor is object-oriented built on top of OpenGL. It comes with a rich set of built-in objects and animation tools. Standard text file format, which is supplied by Open Inventor, is easily used to generate complex objects (such as OEX). Objects are easily added or modified without code modification.

### Future Work

Up to now simulation package has already been used for tuning controller. "Hardware in loop" simulation will be tested in recently. In the coming year, we will improve this simulation software. We will characterize sensors such as GPS/LBL and setup sensors model that will be incorporated into the package. Environment factor will be taken into consider too. Basic currents and waves model will be studied and coded to simulate the environment. More delicate wave and current models will be included with the help of MURI partners. Environment database will also be created. CTD survey data will be incorporated. Feedback adaptive sampling controller development will be

implemented based on the database. Obstacles avoidance algorithm will also be implemented.

### References

Humphreys D. E.(1996) "*Vehicle Hydrodynamic & Maneuvering Model for the FAU Ocean Explorer Vehicle (OEX)*" V.C.T Technical Memorandum 96-05

Ganesan K *et al* (1996) "A *pragmatic software architecture for UUVs*" IEEE Symposium on Autonomous Underwater Vehicle Technology.

An, P. E. *et al* (1996). " A *Quantitative Measure of Sea-State Effect on Small Autonomous Underwater Vehicle Motion in Shallow Water*" , Oceanology International 96, Brighton, UK, pp.211-233.

Fossen, T. I. (1994). *Guidance and Control of Ocean Vehicles*, John Wiley Publishing.

Smith ,S. M. *et al* "The Ocean Explorer AUV: A Modular Platform For Coastal Oceanography" , UUST, Durham, New Hampshire, September.

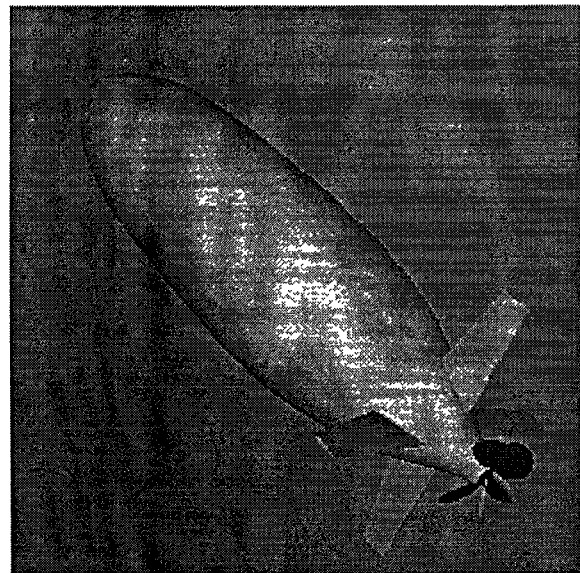


Figure 1 3D AUV model

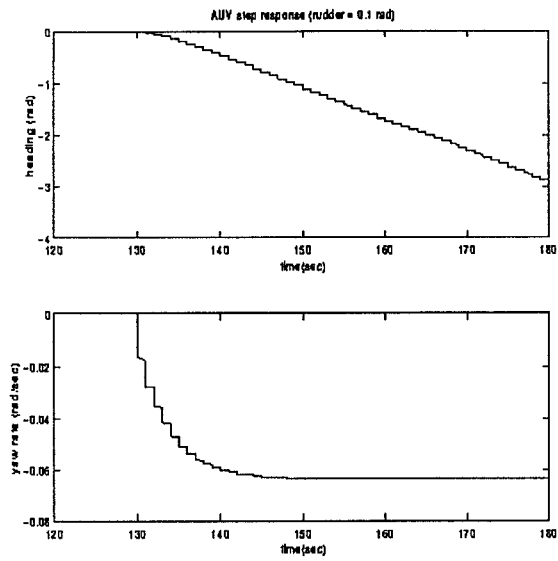


Figure 2 Step response of AUV model

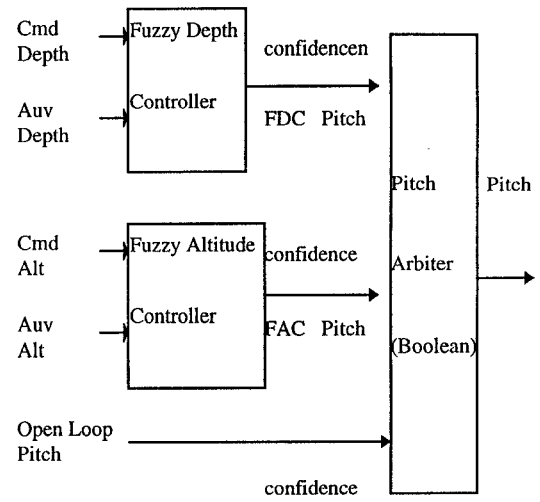


Figure 3 Sketch of manager

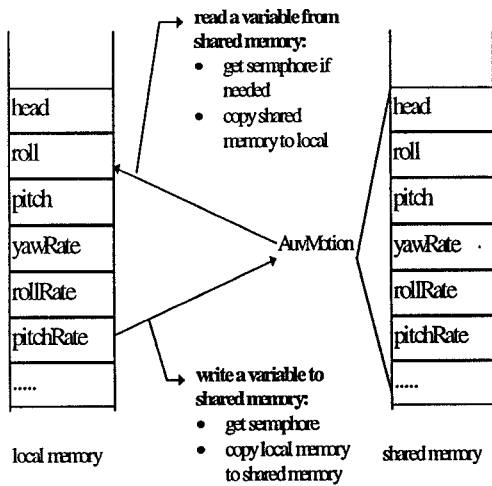


Figure 3 Sketch of shared memory

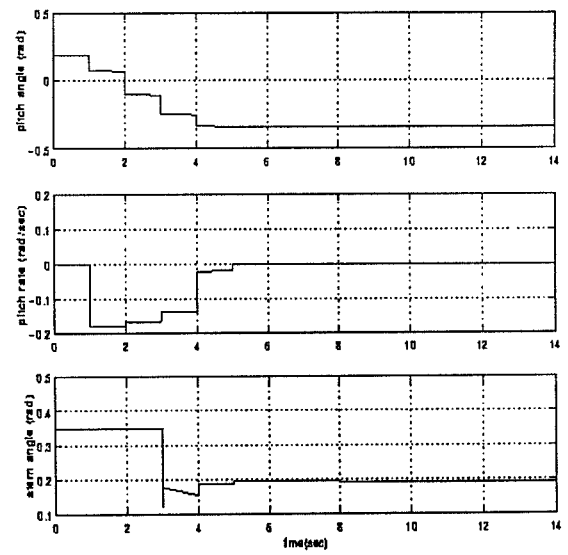


Figure 5 Time history of fuzzy sliding mode pitch controller